图文 022、一步一图：深入揭秘JVM的年轻代垃圾回收器ParNew是如

详情 评论

一步一图

深入揭秘JVM的年轻代垃圾回收器ParNew是如何工作的！

给大家推荐一套质量极高的Java面试训练营课程：

作者是中华石杉，石杉老哥是我之前所在团队的 Leader ，骨灰级的技术神牛！

大家可以点击下方链接，了解更多详情，并进行试听：

21天互联网Java进阶面试训练营（分布式篇）

重要说明：

最近不少同学留言反馈，说希望建立一个微信群，供大家进行JVM专栏的学习交流。

这个提议非常好，不过管理微信群是一件挺费时的事儿，我平时工作较忙，实在抽不出时间来进行群管理.

正好石杉老哥的面试训练营建了微信交流群，并且还请了不少一线大厂的助教。

因此跟石杉老哥商量了一下，决定厚着脸皮“鸠占鹊巢”。购买了我JVM专栏的小伙伴，可以加入石杉老哥的微信群，在群里讨论交流技术。

如何加群，请参见文末（注：如果之前已经加过的，就不要重复加群了）

推荐结束，正文开始

1、前文回顾

上周的文章已经给大家把整个JVM的核心运行原理全部梳理清楚了，大家现在应该对以下问题非常的清晰明了：

对象在新生代分配，然后什么时候会触发Minor GC

触发Minor GC之前会如何检查老年代可用内存大小和新生代对象大小，如何检查老年代可用内存大小和历次Minor GC之后升入老年代的平均对象大小

什么情况下Minor GC之前会提前触发Full GC，什么情况下会直接触发Minor GC

Minor GC之后有哪几种情况对象会进入老年代

而且大家也大概知道了垃圾回收器、垃圾回收线程、垃圾回收算法之间的关系，包括垃圾回收的过程中的“Stop the World”现象和场景对系统运行性能的影响。

这周我们就要来相对深入的研究一下常见的新生代和老年代的垃圾回收器的运行原理了，同时看看常见的垃圾回收参数一般会怎么来设置。

同时结合案例来研究一下在你上线一个新系统的时候，如何通过预估的手段和方法提前对系统的垃圾回收参数进行合理的设置。

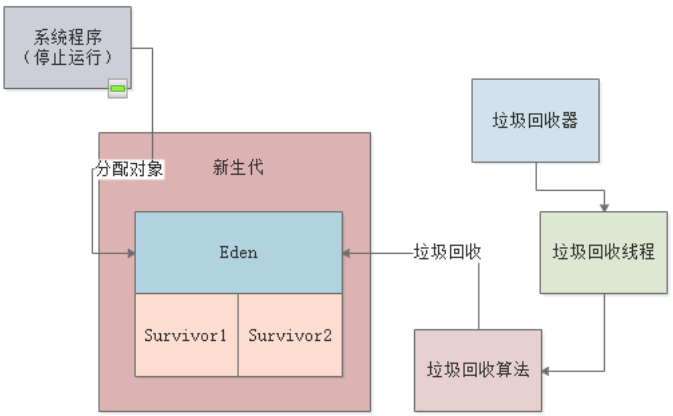
2、最常用的新生代垃圾回收器：ParNew

一般来说，在之前多年里，假设没有最新的G1垃圾回收器的话，通常大家线上系统都是ParNew垃圾回收器作为新生代的垃圾回收器

当然现在即使有了G1，其实很多线上系统还是用的ParNew。

通常运行在服务器上的Java系统，都可以充分利用服务器的多核CPU的优势，所以大家可以想一下，假设你的服务器是4核CPU，如果对新生代垃圾回收的时候，仅仅使用单线程进行垃圾回收，是不是会导致没法充分利用CPU资源？

如下图：

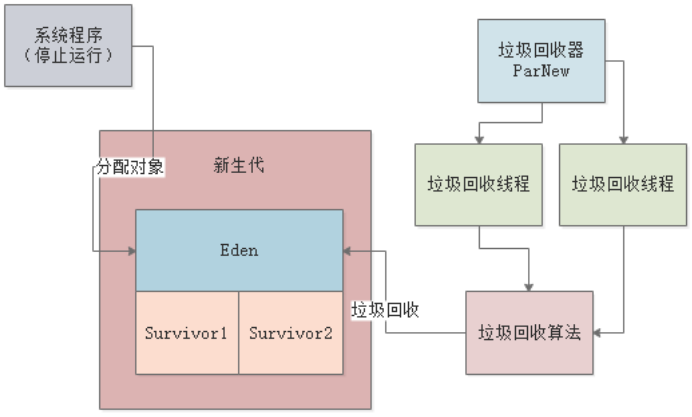


比如上图，现在你在垃圾回收的时候，都把系统程序所有的工作线程全部停掉了，就一个垃圾回收线程在运行

那么此时4核CPU的资源根本没法充分利用，理论上4核CPU就可以支持4个垃圾回收线程并行执行，可以提升4倍的性能！

所以说，新生代的ParNew垃圾回收器主打的就是多线程垃圾回收机制，另外一种Serial垃圾回收器主打的是单线程垃圾回收，他们俩都是回收新生代的，唯一的区别就是单线程和多线程的区别，但是垃圾回收算法是完全一样的。

大家看下图，ParNew垃圾回收器如果一旦在合适的时机执行Minor GC的时候，就会把系统程序的工作线程全部停掉，禁止程序继续运行创建新的对象，然后自己就用多个垃圾回收线程去进行垃圾回收，回收的机制和算法就跟之前说的是一样的。



3、如何为线上系统指定使用ParNew垃圾回收器？

一般来说，对于线上系统部署启动的时候，我们之前都看过多种方式来设置JVM参数了，在Eclipse/IntelliJ IDEA中可以设置Debug JVM Arguments，使用“java -jar”命令启动时直接在后面跟上JVM参数即可

部署到Tomcat时可以在Tomcat的catalina.sh中设置Tomcat的JVM参数，使用Spring Boot也可以在启动时指定JVM参数。

那么在启动系统的时候如果要指定使用ParNew垃圾回收器，是用什么参数呢？

很简单，使用“-XX:+UseParNewGC”选项，只要加入这个选项，JVM启动之后对新生代进行垃圾回收的，就是ParNew垃圾回收器了。

那么Minor GC的时机，检查机制，包括垃圾回收的具体过程，以及对象升入老年代的机制，都是我们之前说过的那套原理了，只不过大家要知道，ParNew会使用多个线程来进行垃圾回收。

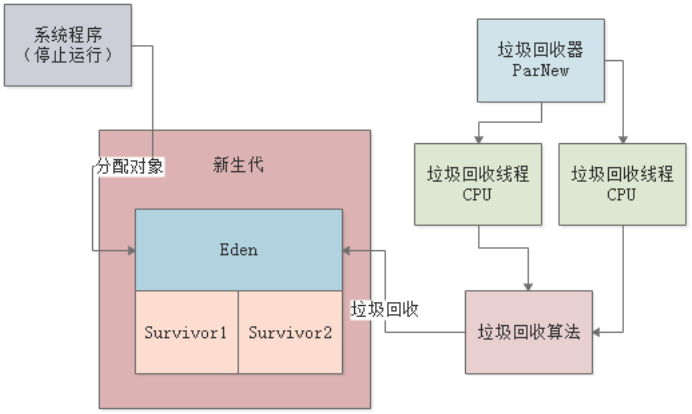
4、ParNew垃圾回收器默认情况下的线程数量

因为现在一般我们部署系统的服务器都是多核CPU的，所以为了在垃圾回收的时候充分利用多核CPU的资源，一旦我们指定了使用ParNew垃圾回收器之后，他默认给自己设置的垃圾回收线程的数量就是跟CPU的核数是一样的。

比如我们线上机器假设用的是4核CPU，或者8核CPU，或者16核CPU，那么此时ParNew的垃圾回收线程数就会分别是4个线程、8个线程、16个线程

这个东西一般不用我们手动去调节，因为跟CPU核数一样的线程数量，是可以充分进行并行处理的。

比如下图，大家可以看到，每个线程都通过一个CPU在运行。



但是如果你一定要自己调节ParNew的垃圾回收线程数量，也是可以的，使用“-XX:ParallelGCThreads”参数即可，通过他可以设置线程的数量

但是建议一般不要随意动这个参数，如果要优化，具体结合后续的案例我们给大家展开。

5、本文总结

这篇文章篇幅不长，主要介绍一下ParNew垃圾回收器，其实垃圾回收器的工作原理之前上周就全部介绍过了

这周的第一篇文章，主要就是对ParNew垃圾回收器本身的多线程原理和相关的参数做一些说明。

6、昨日思考题

上篇文章让大家思考了一个问题，其实反而在这篇文章里要多花点时间来说明。

之前让大家思考的问题就是：

> 到底是用单线程垃圾回收好，还是多线程垃圾回收好？
> 到底是Serial垃圾回收器好还是ParNew垃圾回收器好？

对这个问题要给大家略微展开做点解释。

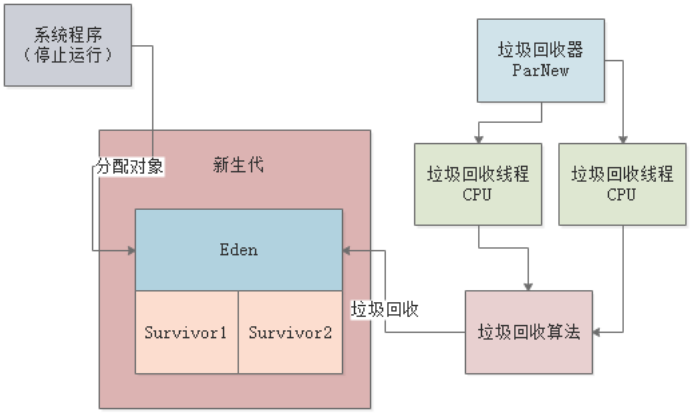
启动系统的时候是可以区分服务器模式和客户端模式的，如果你启动系统的时候加入“-server”就是服务器模式，如果加入“-cilent”就是客户端模式。

他们俩的**区别**就是，如果你的系统部署在比如4核8G的Linux服务器上，那么就应该用服务器模式，如果你的系统是运行在比如Windows上的客户端程序，那么就应该是客户端模式。

那么服务器模式和客户端模式的区别是啥呢？

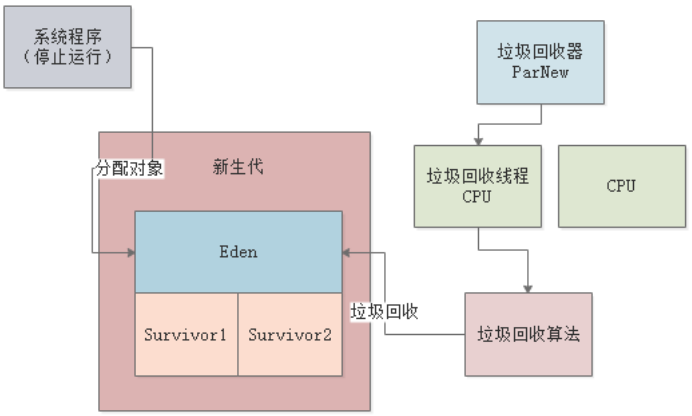
服务器模式通常运行我们的网站系统、电商系统、业务系统、APP后台系统之类的大型系统，一般都是多核CPU

所以此时如果要垃圾回收，那么肯定是用ParNew更好，因为多线程并行垃圾回收，充分利用多核CPU资源，可以提升性能。如下图。



反之如果你部署在服务器上，但是你用了单线程垃圾回收，那么就有一些CPU是被浪费了，根本没用上

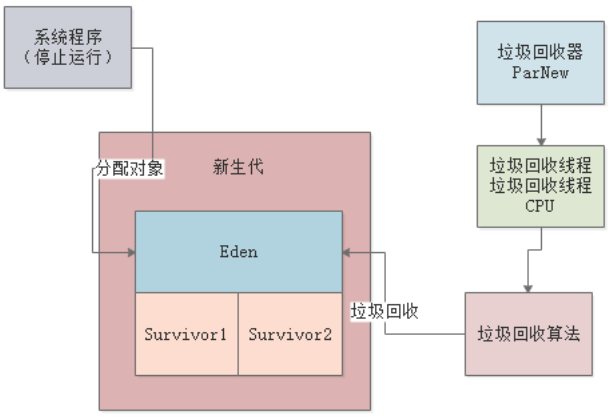
比如下图。



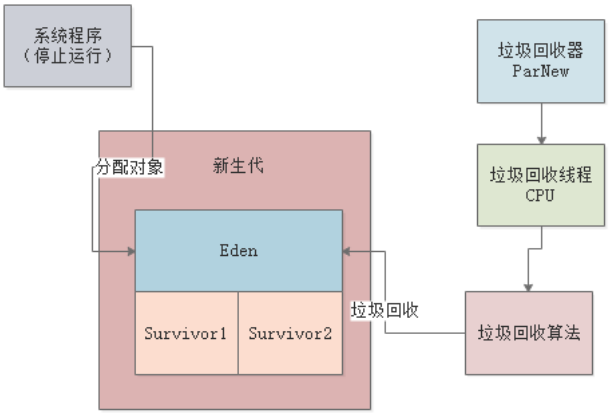
那么如果你的Java程序是一个客户端程序，比如类似百度云网盘的Windows客户端，或者是印象笔记的Windows客户端，运行在Windows个人操作系统上呢?

这种操作系统很多都是单核CPU，此时你如果要是还是用ParNew来进行垃圾回收，就会导致一个CPU运行多个线程，反而加重了性能开销，可能效率还不如单线程好

因为单CPU运行多线程会导致频繁的线上上下文切换，有效率开销，如下图。



所以如果是类似于那种运行在Windows上的客户端程序，建议采用Serial垃圾回收器，单CPU单线程垃圾回收即可，反而效率更高，如下图。



但是其实现在一般很少有用Java写客户端程序的，几乎很少见，Java现在主要是用来构建复杂的大规模后端业务系统的，所以常见的还是用“-server”指定为服务器模式，然后配合ParNew多线程垃圾回收器。

但是大家还是应该清楚单线程和多线程对垃圾回收的适用场景。

7、今日思考题

其实我们一直是鼓励大家在评论区提出有价值的提问的，有自己的思考在里面

昨天有个同学提了一个自己经历过的JVM的面试题，我觉得非常好，给出来让大家思考一下：

“一个面试题，parnew+cms的gc，如何保证只做ygc，jvm参数如何配置?

我的回答（这里指该同学回答）：

加大分代年龄，比如默认15加到30;

修改新生代老年代比例，比如新生代老年代比例改成2:1

修改e区和s区比例，比如改成6:2:2

面试官说他们内部服务已经做到fullgc次数为0，只做ygc，想听听老师的意见

对于这个面试题，如果大家吃透了咱们这个专栏前三周的JVM原理和案例，完全可以把这个面试题回答的滴水不漏

结合案例和画图给面试官说明，而不是干巴巴的简单给几个方法，可以回答的有血有肉，让面试官无话可说。

其实要做到仅仅young gc，而几乎没有full gc是不难的，只要结合自己系统的运行，根据他的内存占用情况，GC后的对象存活情况，合理分配Eden、Survivor、老年代的内存大小，合理设置一些参数，即可做到。